Pattern Recognition, Vol. 23, No. 3/4, pp. 337–346, 1990
Printed in Great Britain

0031-3203/90 $3.00 + .00
Pergamon Press plc

A REVIEW OF SEGMENTATION AND CONTEXTUAL ANALYSIS TECHNIQUES FOR TEXT RECOGNITION*

D. G. ELLIMAN
Drawing Recognition Group, Department of Computer Science, The University of Nottingham, University Park, Nottingham NG7 2RD, U.K.

I. T. LANCASTER
PAFEC Ltd, Strelley Hall, Strelley, Nottingham NG8 9PE, U.K.

(*Received* 23 *January* 1989; *in revised form* 27 *April* 1989; *received for publication* 16 *May* 1989)

Abstract—This paper presents a review of the literature on text-processing techniques. The techniques covered are segmentation of text and contextual recognition, both of which are required when considering text recognition of documents. Several different methods of text segmentation are compared for various image data formats. The problem of correct segmentation of joined and broken characters is also considered. Two techniques for contextual recognition are considered: the Markov-based methods and dictionary look-up methods. The various techniques for storing dictionary information are compared. A discussion of the importance of the choice of the correct context is given, together with guidance on which methods are best suited to which applications.

Text processing Text segmentation Text recognition Markov methods Levenshtein Distance N-gram techniques Dictionary structure Viterbi Algorithm Contextual processing

1. INTRODUCTION

The use of computerized document-handling systems has now become widespread. Applications such as word processors, electronic publishing systems, and computer-aided design systems can be found in many organizations. The installed base of this technology has been growing very rapidly in recent years, and companies are faced with the problem of how to deal with existing documents which are inaccessible to the new systems. The existing methods of using such documents would be to input them manually to the computer, but this can be a time-consuming and laborious task. Documents can be captured in a machine-readable form using a scanner. These products are now available at low cost for A3 and A4 sheets.

The image obtained from this type of scanner is usually in raster format. No information on the primitive symbols making up the document is available. If the document is to be used in conjunction with any of the systems given above then these primitive symbols, whether lines, curves, characters or symbols, must be known. Techniques are therefore required to extract these primitive symbols from the original raster image.

This paper deals specifically with the problem of text recognition from such documents. Any system which aims to automatically recognize text must have the following text-processing capabilities as shown in Fig. 1.[1] Text regions must be segmentated from other regions in the document and further separated into characters and words. It will be necessary to automatically recognize text of different fonts and to verify this recognition using contextual information.

The rest of this paper is divided into four sections (2–5); Section 2 deals with segmentation of text regions from other regions in the document and with the

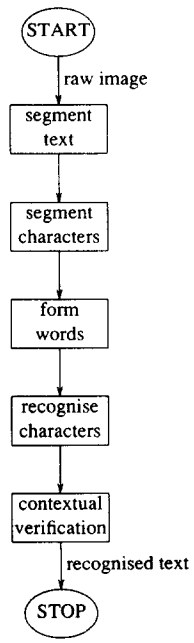

Fig. 1. Flowchart showing process of text recognition.

further segmentation of these regions into individual words and characters. Section 3 investigates methods of contextual text recognition and spelling correction. Section 4 discusses the findings, suggests which methods are best suited to particular applications, and discusses the importance of selecting the correct context. Section 5 contains concluding remarks. Character recognition techniques are not reviewed in this paper but have been extensively reviewed elsewhere.[2-4]

2. TEXT AND CHARACTER SEGMENTATION

A general document will consists of several areas of different image types such as text, diagrams and photographs. Before the text can be recognized at a character or word level it is necessary to identify the text regions in the document. Having identified the text regions these must be further divided to give individual characters and words. At the character level problems may exist in the form of joined or broken characters which are difficult to recognize.

2.1. *Segmentation of text regions*

Text segmentation techniques have been developed for applications such as the processing of documents for facsimile transmission,[5] and the location of post codes on envelopes.[6] Many of these techniques are based on contextual information on the type of image. Such information might be that the text forms horizontal stripes in the image[7] or that characters are all of a similar size and can be discriminated from other drawing information.[8]

Several different approaches to this problem have been used, dependent upon whether the document is represented as a grey-level raster image, a binary raster image or a vector image. Methods applicable to each of these image representations are discussed below.

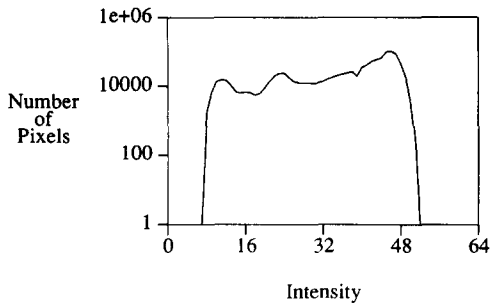

Fig. 3. Intensity histogram for a document containing mixed image types (a photocopier test chart).

2.1.1. *Grey level raster images.* Image properties obtained for different regions in a grey-scale image would appear to be useful in segmenting these regions. Two such properties that might be considered are Fourier spectra and grey-level histograms. Scherl *et al.*[9] investigated both of these properties for different types of image. The Fourier spectra for five different types of image are shown in Fig. 2. From the figure it can be seen that the spectra obtained for the text images are distinct from those of the other image types. It was noted that, for certain fonts, the Fourier spectra could even yield information on the pitch and line spacing.

A histogram for a document containing several different regions is shown in Fig. 3. The histogram is a composite of the histograms for each image region, which can be clearly seen from its overall shape. Scherl *et al.*[9] derived statistics from the histograms of small overlapping areas, but found them incapable of segmenting regions in a general case. The reason for this is that regions of widely differing types can produce very similar histograms.

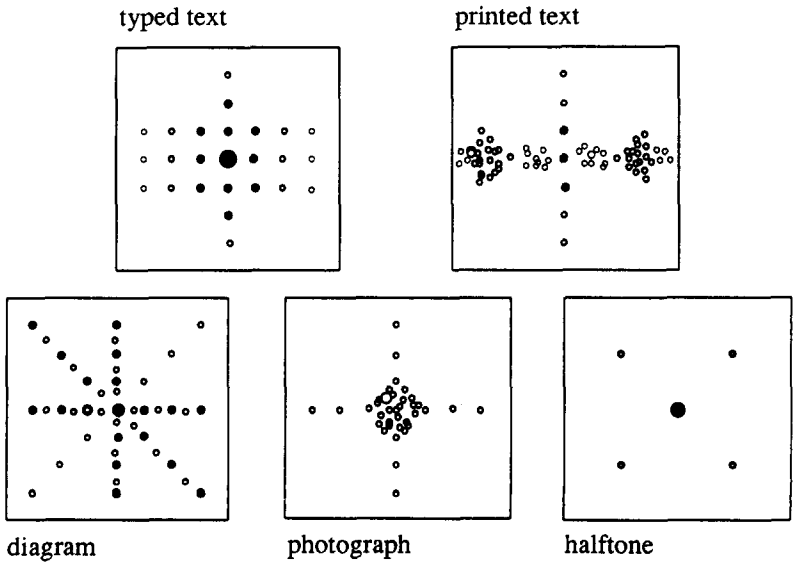

Fig. 2. Schematic representations of Fourier spectra obtained for different image types, after Scherl *et al.*[9]

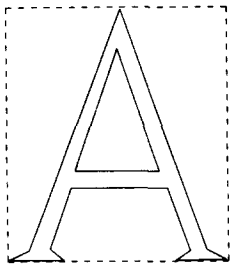

Fig. 4. The minimax box for the letter A.

2.1.2. *Binary raster images.* Two techniques have been considered for segmentation of image regions in binary raster images: runlength-based alogrithms which work at a region level and those which segment individual characters.

The use of runlength-based algorithm was first proposed by Johnston,[7] who noted that on many documents text forms horizontal stripes. Other exponents of this method are Wahl and coworkers.[10,11] Johnston used a grow shrink algorithm to suppress the small white runlengths. The resulting image can be inverted to obtain a mask for segmentation. Pictures tend to become totally fused together and disappear upon inversion. Wahl and coworkers used a constrained runlength algorithm to suppress white runlengths, and thus obtained similar masks to those described above. A linear classifier can then be used to discriminate between regions.

There are two approaches to character-based text segmentation: those based on character size and those based on character recognition. Both of these approaches require that the object pixels are formed into connected regions. Makino[12] and Toyoda *et al.*[8] then segmented a region as text if the size of the minimax box surrounding the region is less than a threshold. An example of a minimax box is shown in Fig. 4. Doster and Schurmann[13] suggested that global features such as the character size do not give good segmentation and proposed a method based on character recognition of the connected regions. A region recognized as a legal character is segmented as such.

2.1.3. *Vector images.* If the image has been vectorized then the document information will be in the form of connected chain vectors. Text can be segmented from this type of data by making use of the fact that characters will tend to be made up of chains comprising a few short vectors. Other context can then be used, such as the maximum or minimum number of endpoints, junctions and loops that occur in legal characters, together with the expected character size.

2.2 *Segmentation of characters*

Many methods of optical character recognition achieve character segmentation by requiring the user to write in the centre of a well-defined box or known locality. This is not applicable to "real-world' documents because characters are often connected to other characters or drawing primitives and can also be disjoint, having broken segments. Two methods of obtaining correct segmentation under these circumstances were found in the literature: segmentation based on pitch estimation or character size and segmentation based on contour analysis.

Nagy and Casey[14] described a system which identifies possible composite characters based on the character width. Tsuji and Asai[15] proposed a system which uses dynamic programming to calculate the character pitch which is thus applicable to both fixed and variable pitch characters. Once the pitch is estimated it can be used to find points for separating joined characters. The character size in the form of a minimax box could also be used to segment broken characters if the broken segment lies within the minimax box.

Shridhar and Badreldin[16,17] reported techniques that can be used to segment and recognize connected and disjoint numeral strings. The recognition and segmentation is based on contour analysis, and recognition rates of 93% are claimed. Numerals are considered to be isolated if a vertical line separates the two numerals. Boken numerals exhibit horizontal or vertical gaps which lead to discontinuities in the contour. Connected numerals are segmented at a decision boundary which is based on the number of transitions from horizontal border to background in a slice across the frame and the smoothness of the left and right contours of the string. Examples of the decision boundaries for typical cases are given in Fig. 5, which also shows the border to background transitions.

2.3 *Word segmentation*

Once characters have been segmented it is necessary to group them into words and phrases if a method of contextual spelling verification is to be used. A method of obtaining such word strings was described by Fletcher and Kasturi.[18] Collinear lines of characters are obtained using a Hough Transform technique. The lines of characters can then be segmented into words using inter-character and inter-word distance information. The pitch estimation technique of Tsuji and Asai[15] would be useful in estimating these distances.

3. STRING VERIFICATION

The need to use contextual information, to verify the results of character recognition, was first identified 30 years ago. This information can take many forms, from the humble word list to word or letter combination probabilities. Three approaches are generally considered for the application of such information in the field of text recognition: Markov methods, dictionary methods and hybrid methods. The Markov methods represent the bottom-up approach, whereas the dictionary methods are top down.[19]

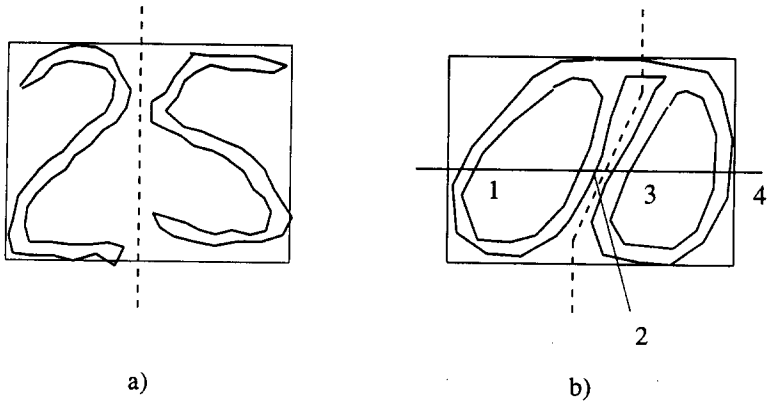

Fig. 5. The decision boundaries for typical cases of character segmentation, after Shridhar and Badreldin,[16] (a) a linear decision boundary for two distinct characters, (b) a non-linear decision boundary for connected characters and the points of border back-ground transition.

3.1. *Markov methods*

These methods model English text as a Markov process which allows transition probabilities to be assigned to various letter combinations or n-grams. The probabilities are usually calculated from the n-gram frequency in a dictionary of legal words, but they can be calculated dynamically from the text being processed. The n-grams can be of any order and the probabilities can be constrained by word position and/or length. Generally, as the order of the n-gram and the number of constraints is increased the accuracy of the results will be improved, but at the cost of computational resources.

These techniques were first used in the field of cryptanalysis. The requirement is to maximize the probability of a given string from all the possible permutations of transition probabilities arising out of the character recognition and thus obtain the most likely result. Several methods have been proposed to maximize this probability including the Viterbi Algorithm,[20-23] the Recursive Bayes Algorithm,[24,25] and probabilistic relaxation.[26] The Viterbi Algorithm has been shown to have complexity $O(nm^2)$, and is essentially parallel in nature, whereas the method of probabilistic relaxation has a complexity of $O(10n^2 + 10)$, where n is the word length and m is the size of the alphabet.

Considerable research into the use of Markov methods in text recognition has been undertaken by Riseman and coworkers,[27-29] who concluded that Markov probabilities did not give significant reductions in word error rate and could even lead to an increased error rate. The reason for this poor performance was that once an error was present it often compounded itself. To overcome this problem Riseman and coworkers considered binary n-grams which can take only one of two values: one if the n-gram is legal and zero otherwise. Word-position-dependent binary trigrams were found to be capable of detecting words in error, fixing the position of the error, and often determining the characters needed to correct the error. Another exponent of binary n-grams, Ullmann,[30] made the point that acceptable results will only be obtained using acceptable resources if special-purpose hardware is available.

3.2 *Dictionary look-up techniques*

Dictionary look-up techniques involve verifying the input word by matching it with a dictionary word. In the simplest form the word will only be verified if it exists in the dictionary. More complex techniques have been developed for approximate string matching, allowing for possible spelling or recognition errors in the input word. The complexity of such techniques depends not only on the string-matching algorithm but also on the method of storing the dictionary.

3.2.1. *String matching.* The aim of string matching is to verify whether two strings are the same or whether one is a misspelling or misrecognition of the other. The classical study of spelling mistakes was undertaken by Damerau,[31] who showed that over 80% of such mistakes fell into one of four error classes involving a single error. These four classes are: one letter wrong, one letter missing, insertion of an extra letter, and transposition of two letters. These errors are generally known as substitution, deletion, insertion and transposition respectively. Most methods proposed for string matching consider a subset of these errors occuring in the input word. Errors occurring from misrecognition or incorrect segmentation are similar to substitution, deletion and insertion errors.

Two approaches are generally considered for the problem of string matching. The first approach is based on an edit distance, which defines the number of edit operations required to obtain the dictionary word from the input word. The second approach uses probabilistic methods to obtain probabilities that the input word is a given dictionary word.

The use of an edit distance was first proposed by Levenshtein,[32] whose method was later modified by Tanaka and coworkers[33,34] to allow weights to be

applied to the edit operations. The use of such weights enables the distance to be modified according to the application. A similar edit distance has been proposed by Wagner and coworkers,[35,36] and a fast algorithm which takes time $O(N^2 \log N)$, if both strings are of length N, has been developed by Masek and Paterson.[37] Such methods have been shown to be capable of correcting deletion, insertion, substitution and transposition errors.

Probabilistic methods for spelling correction have been proposed by Hall and Dowling[38] and Kashyap and Oommen.[39] Probabilistic methods measure the similarity between two strings based on probabilities and likelihoods that one string could be a misrepresentation of the other. Although at first sight the calculation of such probabilities would appear difficult, they lend themselves to recursive or dynamic computation. Computational complexities of $O(lm + ln)$ are claimed, where l is the length of the dictionary, m the average dictionary word length and n the length of the given word. These methods have been shown to be consistent and can correct multiple insertion, deletion and substitution errors.

3.2.2 *Dictionary storage techniques.* The structure of the dictionary is of great importance; it has a large effect on the performance of the verification algorithm, and must therefore allow very fast searches.[40] The earliest considered dictionary structures were generally sequential in nature, requiring the list to be searched up to the word if it exists or to the end if it does not. Many improved methods for dictionary structures have been proposed, three of which will be discussed in this section, namely Trie memory, frequency ordering, and redundant hash addressing.

The idea of Trie memory, which takes advantage of the redundancy of common prefixes, was first introduced by Fredkin.[41] This method of storage allows fast searches in both acceptance and rejection of the given word. The disadvantage of a Trie is that it makes inefficient use of memory when compared with the original word list. There are generally three implementations of this technique, the Trie,[42] the binary tree and the DAWG,[43,44] which are demonstrated in Figs. 6–8.

The DAWG is the most efficient implementation in terms of memory, and could be further improved by replacing common suffixes such as "ION", "ING" and "ED" with a single node. Knuth[42] has shown that the Trie will take $O(\log_2 N/\log_2 M)$ iterations on average and will require $O(MN/\ln M)$ memory, where N is the size of the dictionary and M the order of the Trie. The computational requirement of the Trie is therefore less than that for the binary tree which requires $O(\log_2 N)$ comparisons.

Other methods have been proposed, which make use of the skewed frequency distribution of the English language. The classical solution to this problem is the frequency-ordered binary search tree as shown in Fig. 9; however, this can become unbalanced if the

7		CAB		CAD		
6	7					
5	BEAD			BED		BE
4					BADE	BAD
3				4		
2	3				5	
1	A	2	6	DAB		
	A	B	C	D	E	_

Fig. 6. A Trie for the words DAB, BAD, BADE, BE, BED, BEAD, CAB, CAD, and A, after Knuth.[42]

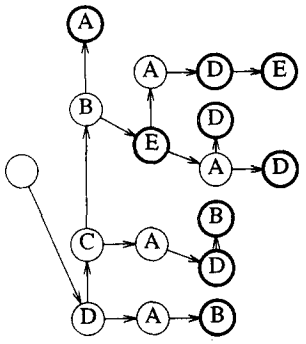

Fig. 7. A Trie for the words DAB, BAD, BADE, BE, BED, BEAD, CAB, CAD, and A represented as a linked list or binary tree.

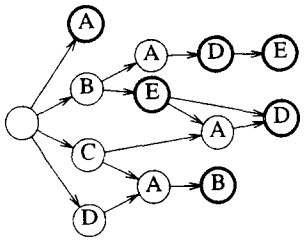

Fig. 8. A DAWG for the words DAB, BAD, BADE, BE, BED, BEAD, CAB, CAD, and A.

frequency ordering is confounded with the dictionary ordering. Sheil[45] described how the use of median split trees can overcome this problem and result in a perfectly balanced tree. A median split tree is a split tree in which a nodes split value is the median, with respect to lexical ordering, of the node values of its descendants. An example of a median split tree is shown in Fig. 10. The costs of a successful search using the median split tree is 3.127 compared with 4.042 for the frequency-ordered binary tree.

Kohonen and Reuhkala[46] proposed a method of dictionary storage based on redundant hash addressing. A hash index is built into the dictionary based on an index calculated from a hash function for each word. The hash function does not give unique

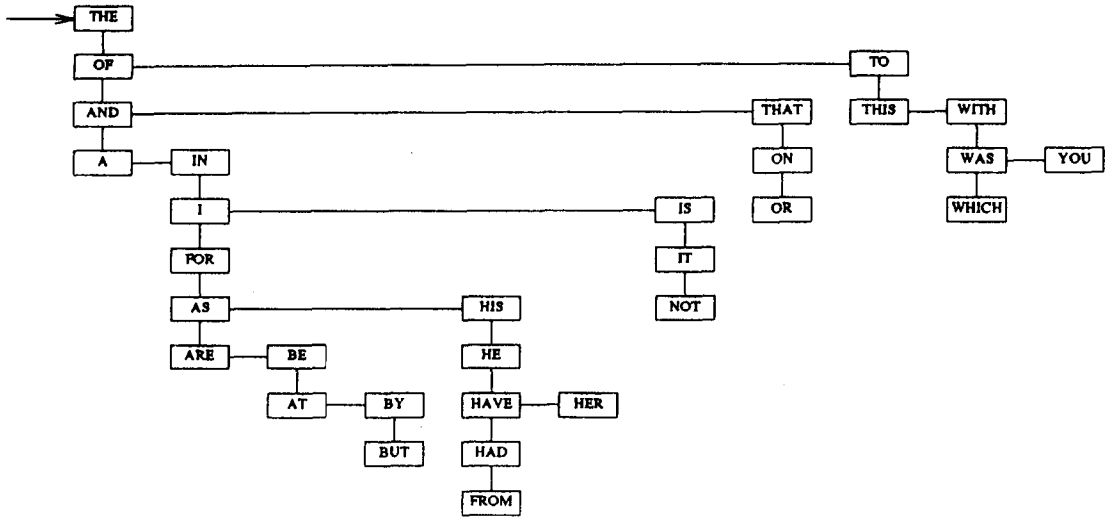

Fig. 9. A frequency-ordered binary tree for the 31 most common English words, after Sheil.[45]

addresses for each word, but should associate similar words with the same address.

3.3 *Hybrid methods*

Hybrid methods for text recognition combine the top-down and bottom-up approaches. Shinghal and Toussaint,[47] Shinghal,[48] Hull and coworkers[49,50] and Sinha and Prasada[51] described methods for applying such an approach. Markov techniques are used, in the form of the Viterbi Algorithm, to form likely alternatives to the input word, which are then searched for in a dictionary. If the word is not found in the dictionary Markov techniques can be used to obtain the most likely result.

The advantage of such hybrid methods is that high-order Markov dependencies do not have to be introduced because accuracy is ensured by using the dictionary. Such techniques therefore reduce the computational complexity of the problem without sacrificing performance.

4. DISCUSSION

The implementation of any text recognition technique is dependent upon the particular application. The application defines the allowable level of misclassification, which may be less critical in data compression than document recognition for example. The available image data format and results from other processes, such as character recognition, will also influence the implementation. Although some techniques may be applicable only to certain image formats it may be possible to convert the image to the correct format. The character recognition may return a choice of characters with associated probabilities which would be useful when considering string matching techniques.

The following sections discuss and compare the text processing techniques outlined above. The advantages and disadvantages of each method are identified, together with their suitability for a particular application.

4.1. *Text and character segmentation*

The Fourier spectra technique appears to be the most promising method for segmenting text in grey-scale images. Much information, such as text orientation, size, and pitch can be obtained from such spectra. Excessive processing time may, however, be involved with such techniques if specialist hardware is not available.

The runlength-based methods, used to segment text in binary images, are very sensitive to changes in text angle and size. Such methods perform badly when line spacing is small and are incapable of segmenting text within diagrams. The use of a method based on character size will overcome these problems, but is still sensitive to changes in character size. The performance of a method based on character size will be very dependent upon the image, esepcially if there are other similar sized primitives. Similar comments to those given for the methods based on character size may also apply to the vector-based method.

The most general technique is that based on character recognition. It could be applied to both raster and vector images, dependent upon the method of character recognition used. All connected regions must be processed, which may result in excessive computational effort. This method will not be sensitive to differing character size, but may result in misclassification of non-text primitives.

The methods of segmentation of joined and broken characters appear to be *ad hoc* in nature and thus not applicable to general documents. An example of this is the contour analysis method of Shridhar and Badreldin, which requires *a priori* knowledge of the number of numerals in the string. Bornat and Brady[52] made the astute observation that reliable

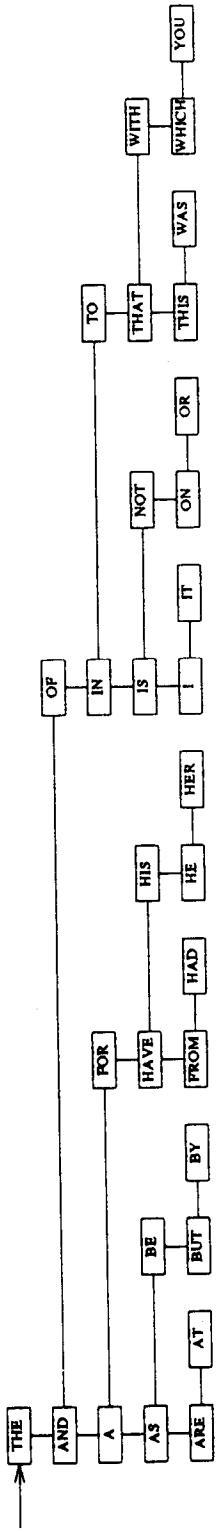

Fig. 10. A median split tree for the 31 most common English words, after Sheil.[5]

segmentation is possible only when some initial hypotheses on character identity have been proposed. This shows that the character segmentation is a "chicken and egg" situation which would probably be best solved using some form of flexible problem-solving strategy such as a blackboard with knowledge sources.[53] The problem appears almost impossible if approached in an entirely bottom-up manner.

4.2 *String verification*

The dilemma of using n-gram probabilities for word verification is the trade-off between computer resources and accuracy. To overcome the problem of ambiguous n-grams for the same error, n-grams of higher order or greater dependency need to be considered. The use of such n-grams can, however, result in prohibitive storage requirements.

A discussion of n-gram statistics, relating to text processing applications, was presented by Suen.[54] The statistics were calculated from several corpi of English language texts and were shown to be more reliable for frequently occurring words. Suen concluded that, to make effective use of n-gram probabilities, the context should be specific to the application.

String matching techniques based on the Levenshtein Distance and its derivatives have been shown to give good results for long words, but often return poor results for short words. The probabilistic methods, are, however, equally applicable to both long and short words. Great improvements in accuracy have been obtained by using probabilistic methods for short words instead of the Levenshtein Distance.

The computational performance of a string matching algorithm depends not only upon the algorithm chosen but also upon the dictionary structure used. The optimum storage method will therefore result from a trade-off between processing time and memory requirements. The Trie/DAWG is the best compromise with regard to processing time as it allows fast searches both when the word exists in the dictionary and when it does not. Other methods may be faster for one or other of the cases, but not for both.

The problem of storage can be overcome by making use of the skewed nature of the frequency distribution for the English language. Half of an average text can be accounted for by the 127 most frequent words. These most common words can be held as a DAWG in core memory which is always searched first. The rest of the dictionary can be held in a different format as a file which is searched only if the word is not found in the DAWG. Sinha[55] has developed a series of factors which allow the design of such a hybrid storage technique to be optimized.

The advantage of dictionary methods is their accuracy; the word either exists in the dictionary or it does not. This accuracy is reduced if string matching is used to correct spelling or recognition errors because a choice of words could be returned in ambiguous cases. The disadvantage of these methods is their slowness. The advantage of the Markov methods is their speed, but their accuracy is questionable because a word is not verified only asserted to be a probable. Both methods will require large amounts of memory to hold the contextual information. Hybrid methods have been developed to optimize the advantages of both methods. The optimum performance, as a trade-off between resources and recognition, must therefore come from a hybrid method.

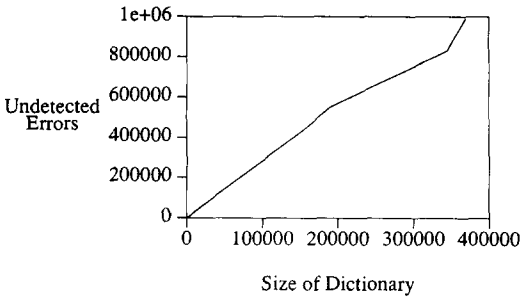

Fig. 11. Increase in undetected typing errors with increase in the size of the dictionary, after Peterson.[57]

4.3 *Choice of correct context*

A source of contextual information is required, whether considering Markov techniques, dictionary look-up techniques, or a combination of both. This source usually takes the form of a dictionary or corpus. Dictionary methods use the information directly, whereas Markov methods use it indirectly to obtain transition probabilities.

The obvious answer to obtaining a dictionary would be to buy one "off the shelf". This, however, has several problems:[56] machine readable dictionaries are often scarce and expensive; those that are available will not contain all forms of a word; a large dictionary will contain many archaic, esoteric or technical words;[40] a general dictionary will contain many similar words and often some spelling errors or American/English spellings.

The choice of correct context is therefore very important. The use of large dictionaries will result in a substantial number of undetected spelling errors[57,58] because a misspelt word is more likely to match another legal word, as demonstrated in Fig. 11. The best dictionary or corpus for a particular application must be one derived from the types of document to be checked.[59]

When considering large dictionaries the search time will be proportionally longer than that for a small specific dictionary. If an application is very specific the words which must be recognized may not appear even in a large proprietary dictionary. A good compromise is to use a small self-learning dictionary which is specific to the application in hand, but can be updated when new words are found.

5. CONCLUSIONS

The choice of character segmentation method will be dependent upon the available image format and

the particular application. The character recognition method is probably the most general technique.

No truly general method of character segmentation has yet been developed. The interpretation of each putative character is mutally dependent on those in its neighbourhood. There is need for a method which selects the optimum solution, given model-based constraints and recognition probabilities for individual characters.

The best technique for string verification is a hybrid method combining n-gram probabilities to obtain likely misspellings or misrecognitions and a dictionary, the most frequent words of which are held as a Trie/DAWG.

To obtain the optimum accuracy of the results, careful attention must be paid to the choice of context. Results obtained will be more accurate if the context is specific to the application.

Acknowledgement—This work was supported by SERC and PAFEC Ltd under grant no. GR/E 42075.

REFERENCES

1. K. Y. Wong, R. G. Gasey and F. M. Wahl, Document analysis system, *IBM J. Res. Dev.* **26**(6), 647–656 (1982).
2. C. Y. Suen, M. Berthod and S. Mori, Automatic recognition of handprinted characters — the state of the art, *Proc. IEEE* **68**(4), 469–487 (1980).
3. C. Y. Suen, Character recognition by computer and applications, *Handbook of Pattern Recognition and Image Processing*, T. Y. Young and K. S. Fu, Eds, pp. 569–586. Academic Press, New York (1986).
4. J. Mantas, An overview of character recognition methodologies, *Pattern Recognition* **19**(6), 425–430 (1986).
5. W. K. Pratt, P. J. Capitant, W. H. Chen, E. R. Hamilton and R. H. Wallis, Combined symbol matching facsimile data compression system, *Proc. IEEE* **68**(7), 786–796 (1980).
6. P. S. Yeh, S. Antoy, A. Litcher and A. Rosenfeld. Address location on envelopes, *Pattern Recognition* **20**(2), 213–227 (1987).
7. E. G. Johnston, Printed text discrimination, *Comput. Graphics Image Processing* **3**, 83–89 (1974).
8. J. Toyoda, Y. Naguchi and Y. Nishimura, Study of extracting Japanese newspaper article, *Proc. 6th Int Conf. Pattern Recognition*, pp. 1113–1115 (1982).
9. W. Scherl, F. Wahl and H. Fuschberger, Automatic separation of text, graphic and picture segments in printed material, *Pattern Recognition in Practice*, E. S. Gelsema and L. N. Kanal, Eds., pp. 213–221. (1980).
10. L. Abele, F. Wahl and W. Scherl, Procedures for an automatic segmentation of text, graphic and halftone regions in documents, *Proc. 2nd Sandinavian Conf. Image Analysis*, pp. 177–182 (1981).
11. F. M. Wahl, K. Y. Wong and R. G. Casey, Block segmentation and text extraction in mixed text/image documents, *Comput. Graphics Image Processing* **20**, 375–390 (1982).
12. H. Makino, Representation and segmentation of document images. *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pp. 291–296 (1983).
13. W. Doster and J. Schurmann, A step towards intelligent document input to computers, *Proc IEEE Conf. Computer Vision and Pattern Recognition*, pp. 515–516 (1983).
14. R. G. Casey and G. Nagy, Recursive segmentation and classification of composite character patterns, *Proc. 6th Int. Conf. Pattern Recognition* Vol. **2**, pp. 1023–1025 (1982).
15. Y. Tsuji and K. Asai, Character image segmentation, *Proc. Soc. Photo-opt. Instrum. Engrs* **504**, 2–9 (1984).
16. M. Shridhar and A. Badreldin, Recognition of isolated and simply connected handwritten numerals, *Pattern Recognition* **19**(1), 1–12 (1986).
17. M. Shridhar and A. Badreldin, Context-directed segmentation algorithm for handwritten numeral strings, *Image Vision comput.* **5**(1), 3–9 (1987).
18. L. A. Fletcher and R. Kasturi, A robust algorithm for text string separation from mixed text/graphics images. *IEEE Trans. Pattern Analysis Machine Intell.* **10**(6), 910–918 (1988).
19. G. T. Toussaint, The use of context in pattern recognition, *Pattern Recognition* **10**, 189–204 (1978).
20. A. J. Viterbi, Error bounds for convolutional codes and an asymptomatically optimum decoding algorithm, *IEEE Trans. Information Theory* **13**(2), 260–269 (1967).
21. G. D. Forney, The Viterbi Algorithm, *Proc. IEEE* **61**(3), 268–278 (1973).
22. D. L. Neuhoff, The Viterbi Algorithm as an acid in text recognition, *IEEE Trans. Information Theory* **21**, 222–226 (1975).
23. R. Shinghal and G. T. Toussaint, Experiments in text recognition with the modified Viterbi Algorithm, *IEEE Trans Pattern Analysis Machine Intell.* **1**(2), 184–193 (1975).
24. J. Raviv, Decision making in Markov chains applied to the problem of pattern recognition, *IEEE Trans. Information Theory* **3**(4), 536–551 (1967).
25. R. Shinghal, D. Rosenberg and G. T. Toussaint, A. simplified heuristic version of a recursive Bayes algorithm for using context in text recognition, *IEEE trans. Systems Man Cybernetics* **8**(5), 412–414 (1975).
26. A. Goshtasby and R. W. Ehrich, Contextual word recognition using probabilistic relaxation labelling, *Pattern Recognition* **21**(5), 455–462 (1988)
27. E. M. Riseman and R.W. Ehrich, Contextual word recognition using binary diagrams, *IEEE Trans. Comput.* **20**(4), 397–403 (1971).
28. E. M. Riseman and A. R. Hanson, A contextual postprocessing system for error correction using binary n-grams *IEEE Trans. Comput.* **2**(5), 480–493 (1974).
29. A. R. Hanson, E. M. Riseman and E. Fisher, Context in word recognition, *Pattern Recognition* **8**, 35–45 (1976).
30. J. R. Ullman, A binary N-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words, *Comput. J.* **20**, 141–147 (1977).
31. F. J. Damerau, A technique for computer detection and correction of spelling errors, *Commun. ACM* **7**(5), 171–176 (1964).
32. V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, *Sov. Phys. Dokl.* **10**(8), 707–710 (1966).
33. T. Okuda, E. Tanaka and T. Kasai, A method for the correction of garbled words based on the Levenshtein Metric, *IEEE Trans. Comput.* **25**(2), 172–178 (1976).
34. E. Tanaka and T. Kasai, Synchronisation and substitution error correcting codes for the Levenshtein Metric, *IEEE Trans. Information Theory* **22**(2), 172–178 (1976).
35. R. A. Wagner and M. J. Fischer, The string to string correction problem, *J. ACM* **21**(1), 168–173 (1974).
36. R. Lowrance and R. A. Wagner, An extension of the string to string correction problem, *J. ACM* **22**(2), 173–183 (1975).
37. W. J. Masek and M. S. Paterson, A faster algorithm computing string edit distance, *J. Comput. System. Sci.* **20**(1), 18–31 (1980).
38. P. A. V. Hall and G. R. Dowling, Approximate string matching, *ACM Comput., Surv.* **12**(4), 381–401 (1980).
39. R. L. Kashyap and B. J. Oommen, Spelling correction using probabilistic methods, *Pattern Recognition Lett.* **2**(4), 147–154 (1984).
40. J. L. Peterson, *Computer Programs for Spelling Correction*. Springer-Verlag, Berlin (1980).

41. E. Fredkin, Trie memory, *Commun. ACM* **3**(9), 490–5000 (1960).
42. D. E. Knuth, Digital searching, *The Art of Computer Programming*, Vol. 3, pp. 481–4999. Addison-Wesley, Reading, MA (1973).
43. A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M. T. Chen and J. Seiferas, The smallest automaton recognising the subwords of a text, *Theor. Comput. Sci.* **40**, 31–55 (1985).
44. A. W. Appel and G. J. Jacobson, The world's fastest scrabble program, *Commun. ACM* **31**(5), 572–579 (1988).
45. B. A. Sheil, Median split trees: a fast lookup technique for frequently occurring keys, *Commun. ACM* **21**(11), 947–958 (1978).
46. T. Kokonen and E. Reuhkala, A very fast associative method for the recognition and correction of misspelt words, based on redundant hash addressing, *Proc. 4th Int. Joint Conf. Pattern Recognition*, pp. 807–809 (1978).
47. R. Shinghal and G. T. Toussaint, A bottom-up and top-down approach to using context in text recognition, *Int. J. Man–Machine Stud.* **11**, 201–212 (1979).
48. R. Singhal, A hybrid algorithm for contextual text recognition, *Pattern Recognition* **16**(2), 261–267 (1983).
49. S. N. Srihari, J. J. Hull and R. Choudhuri, Integrating diverse knowledge sources in text recognition, *ACM Trans. Office Information sys.* **1**(1), 68–87 (1983).
50. J. J. Hull, S. N. Srihari and R. Choudhuri, An inegrated algorithm for text recognition: comparison with a cascaded algorithm, *IEEE Trans. Pattern Analysis Machine Intell.* **5**(4), 384–395 (1983).
51. R. M. K. Shinha and B. Prasada, Visual text recognition through contextual processing, *Pattern Recognition* **21**(5), 463–479 (1988).
52. R. Bornat and J. M. Brady, Using knowledge in the computer interpretation of handwritten FORTRAN coding sheets, *Int. J. Man–Machine Stud.* 8, 13–27 (1976).
53. L. D. Erman, F. Hayes-Roth, V. R. Lesser and D. R. Deddy, The Hearsay-II speech understanding system: integrating knowledge to resolve uncertainty, *Comput. Surv.* **12**(2), 213–253 (1980).
54. C. Y. Suen, n-Gram statistics for natural language understanding and text processing, *IEEE Trans. Pattern Analysis Machine Intell.* **1**(2), 164–172 (1979).
55. R. M. K. Sinha, Some characteristic curves for dictionary organisation with digital search, *IEEE Trans. Syst. Man Cybernetics* **17**(3), 520–527 (1987).
56. T. N. Turba, Checking for spelling and typographical errors in computer-based text, *ACM SIGPLAN-SIGOA Newslett.* pp. 51–60 (1981).
57. J. L. Peterson, A note on undetected typing errors, *Commun. ACM* **29**(7), 633–637 (1986).
58. M. D. McIlroy, Development of a spelling list, *IEEE Trans. Commun.* **30**(1), 91–99, (1982).

About the Author—IAN LANCASTER received the B.Sc. degree in Mechanical Engineering from the University of Birmingham in 1980 and is a Member of the British Computer Society. He worked for Marconi Space and Defence Systems Ltd and National Nuclear Corporation before joining PAEFEC Ltd as a programmer in 1984. He currently holds the position of Senior Programmer and is a member of the RAVEN development group. He is seconded to Nottingham University as part of a project, on recognition of engineering drawings, jointly funded by PAFEC and SERC. He is studying part time for a Ph.D. degree in Computer Science.

About the Author—DAVID ELLIMAN received the B.S.c. in Mechanical Engineering from the University of Nottingham in 1973. He then joined the research staff of Rolls-Royce and developed computer models of flow and combustion processes, and was awarded a Ph.D. degree in 1977. Subsequently, he was employed designing CAD/CAM workstations at Quest Automation Ltd, as a director of a software house, and as a consultant. He was appointed a lecturer in software engineering in 1983, and has research interests in computer vision, expert systems, CAE and formal methods. Dr. Elliman is a chartered member of the IEE, and BCS and the BPRA.